Data Modeling

Session 2 Brock Tibert

Related to last week:

Concepts and Intuition > Tools

ExamSoft

- I plan on using ExamSoft for certain portions of this course
- There is a getting started (ungraded) assessment associated with this class that closes 9/17
- Please use that to ensure that you can successfully install and complete the assessment
- Student ExamSoft guide
- questromhelp@bu.edu for help/assistance

Today's Agenda

- Review considerations and intuition about data movement and integration (data integration/pipelines)
- Use above to help reason about schema design (data modeling)
- Demos:

- LucidChart for ERD Design
- Get back into BigQuery (brief GCP refresher, DDL/DML)

- Hands-on open-ended practice:

- In-class Project for the semester
- Schema design

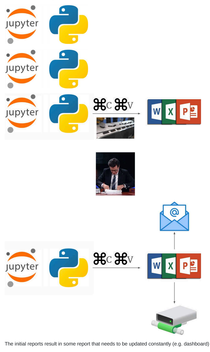
Common “Analytics Pipeline”

DA/DS on a “client” engagement where

your firm delivers: 1. Initial insights @ “client” meeting

Clients can be internal or external

2. Operationalize insights for your partner



The initial reports result in some report that needs to be updated constantly (e.g. dashboard)

Analytics Flow - Data Review 2

Our response when our boss sends us a Slack

message asking about Dashboard updates for review 4.....

that we haven't ran yet.

What could go wrong?

Initial Analytics Work: Inevitability Kicks In

Peer Group Think: 3 Minutes

What could be sources of error in this flow?

- If API, limits, access

permissions changed ● Server issues 5XX ● Deliveries modified by holidays ● Changing filetypes ● Filenaming issues (XL -> PDF) ● Columns named changed ● Package version changed (pip

install -U)

●

Errors Happen, but (resilient) pipelines can help us

- If we think backwards, how can we ensure our end users always have access to some data

- It just may not be the most up to date, but some is better than no reports

- We need to go beyond pd.read_csv
- We need to think about the data/metadata we are generating
- What about data that has changing definitions or values?
- How/where are we storing this data

Segue to:

- Different ways to think about working with data feeds
- How we can/should store data (not on our laptop!)
- Breaking apart monolithic jobs into smaller tasks
- Considerations/requirements on data freshness and patterns for updates

Top-down: Data Pipelines

Data Integration Overview

ETL, ELT, EtLT

- Extract

- Query data from the application(s). The results of the query are what get used downstream.

- Transform

- Modify the extracted data to confirm to business rules, cleaning the data, and to be modeled for the warehouse. This happens 'in flight'.

- Load

- Insert the data into the warehouse table(s)

- Variants

- EtLT = extract the raw data from the application(s) and load the data into the warehouse unmodified, in order to retain the original form before transforming the data as it resides in the warehouse. This is becoming increasingly popular with data teams.
- EtLT = extract the raw data, perform some small transformations in flight, load the data into the warehouse, and then perform the bulk of the transformations.

ETL and ELT Framing

Peer Thought Exercise:

What are some design considerations you need to keep in mind when deciding whether to use ELT versus ETL, particularly concerning the types of transformations required and the size of the data being processed?

● Determine a data feed/integration where each strategy

would make sense.

● What are some of the properties where one approach

might be better than another

● Are there 'tricks' we can use to offset some of the

challenges of a given approach?

Quick Refresher: Cloud/Object Storage

- Store files with high availability (think: limitless hard drive)
- Storage is cheap, and can be reduced further with archival settings
- Can be used for all sorts of needs

- Data
- Logs
- Application/database backups
- Media storage
- Compliance requirements

- Foundation for data lakes/houses, but we need additional tooling to impose structure (schemas) on the data
- Design considerations are important

- Bucket partitioning
- File formats

Cloud storage as a landing zone

Quick Aside: Data Lakes

Cloud Data Warehouse

- Structured Storage for Analytics
- Fast. Really fast.
- Petabyte scale data.
- SQL interface (variants exist)
- ML(ish) inside the warehouse at query runtime
- Security controls for access (Data Scientists see certain views, Analysts another set of views)
- Some not-so-obvious facts about BigQuery/Redshift

- Primary Keys are not enforced as value constraints. Indexes used for performance
- As your data grows, you need to consider how the data are stored in your tables relative to access patterns
- Can use BQ/RS as an interface on top of your files in cloud storage.
- RS query optimization behind the scenes

Data Pipeline Considerations

Data Warehouse (Our Focus)

- We can get performance gains from the storage in the warehouse
- Depending on the provider, costs can vary but can be (very) expensive (Redshift).

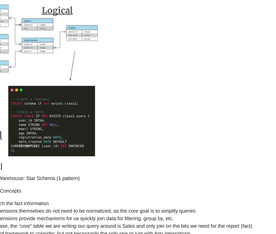
- Pricing models are starting to change (separation of storage and compute/data scans)

- While BQ/RS can query data in object storage, it may not be as performant as querying the data inside the data engine
- Tuning can help costs and performance

Data Lake

- Can be less expensive to start
- Requires metadata layers on top of the data (Data Catalogs) to support queries
- As noted before, careful consideration of file formats (parquet common) and storage (partitions) for data scans.
- We typically aren't doing things like select * from table blc the data are being assembled on the fly via the query execution

Ok, that's a lot. Let's zoom out!



Quick Group Reflection

What might you have done differently in some of your team projects here at Questrom based on the pipelines we just discussed? Could moving beyond flat files helped?

We are in a safe space!

Zoom In: Core skills

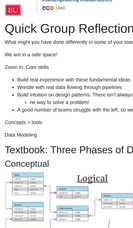
- Build real experience with these fundamental ideas
- Wrestle with real data flowing through pipelines
- Build intuition on design patterns. There isn't always
  - no way to solve a problem!
- A good number of teams struggle with the left, so we are going to focus here

Concepts -> tools

Data Modeling

Textbook: Three Phases of Data Modeling

Conceptual



Physical

App -> Data Warehouse: Star Schema (1 pattern)

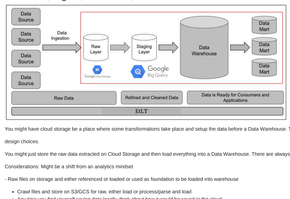
Star Schema Concepts

- We enrich the fact information
- The dimensions themselves do not need to be normalized, as the core goal is to simplify queries
- The dimensions provide mechanisms for us quickly join data for filtering, group by, etc.
- In this case, the "core" table we are writing our query around is Sales and only join on the bits we need for the report (fact)
- Pattern of framework to consider, but not necessarily the only one or just with App integrations

Important: There isn't one way! Guidelines on how to think about data storage, cleanliness, and backups! Different tools play in different sections of the

diagrams and flows we saw. There are always tradeoffs in design choices.

Σ: Collect, Ingest, Transform, Present for End Users



You might have cloud storage be a place where some transformations take place and setup the data before a Data Warehouse. There are always tradeoffs in

design choices.

You might just store the raw data extracted on Cloud Storage and then load everything into a Data Warehouse. There are always tradeoffs in design choices.

Considerations: Might be a shift from an analytics mindset

- Raw files on storage and either referenced or loaded or used as foundation to be loaded into warehouse

- Crawl files and store on S3/GCS for raw, either load or process/parse and load
- Any time you find yourself saving data locally, think about how it would be saved in the cloud

- We can leverage database schemas to organize the flows/work

- Raw/source schema for the files loaded (or light transforms)
- Staging tables for processing and intermediate tasks/transformations
  - Intuition: DataFrames you create during your analysis to help downstream tasks
- Curated/processed/finished tables/views where end users start to access
- SQL to move data between schemas keeping the data inside the data

- Database/Warehouse tables are the original DataFrames!
- Not every data processing flow needs to follow the same pattern

- Need to consider the number of teams/domains that need to be supported
- Always start small! Careful consideration for future expansion but no need to do everything all at once. Iteration on data models is normal!

Quick Group Reflection -

Transformations

● Can you think of scenarios why SQL-based transformations are beneficial/necessary? ● Can you think of scenarios

where SQL transformations may not meet our needs? (strict ELT)

● What about new data. How

might that impact the flow?

●